COMPARAÇÃO DE INTELIGÊNCIAS ARTIFICIAIS GENERATIVAS NA REFATORAÇÃO DE CÓDIGOS

Walcileine Larissa Kleinschmidt[1], Giácomo Antônio Althoff Bolan[2]

Resumo: Este estudo aborda o uso de inteligências artificiais generativas na refatoração de código, prática essencial para melhorar a legibilidade e a manutenção de sistemas. O objetivo geral foi comparar o desempenho de cinco ferramentas gratuitas, ChatGPT, GitHub Copilot, Bing, Gemini e Perplexity em tarefas de refatoração de trechos com diferentes níveis de complexidade nas linguagens Python, Java, JavaScript e C#. A pesquisa seguiu metodologia qualitativa comparativa: realizou-se revisão bibliográfica em bases como Scopus, IEEE Xplore e Google Scholar (2018–2024), Foi selecionado 5 códigos dos repositórios públicos do GitHub e aplicou-se prompt padrão para todas as IAs. As refatorações foram avaliadas por profissionais com mais de três anos de experiência, usando escala Likert de 1 a 5. Os resultados mostraram que o desempenho das IA's em cada linguagem está diretamente relacionado à base de dados utilizada em seu treinamento. Conclui-se que a escolha da IA ideal depende da linguagem e do contexto.

Palavras-chave: Refatoração; Inteligencia Artifical Generativa;Avaliação comparativa; ChatGpt; Perplexity; Gemini; GitHub; Bing.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), walcileine@gmail.com
[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), kinhobolan@live.com

ABSTRACT: This study examines the use of generative artificial intelligence tools for code refactoring, a practice essential to improving code readability and maintainability. The primary objective was to compare the performance of five free platforms, ChatGPT, GitHub Copilot, Bing, Gemini, and Perplexity, on refactoring tasks of varying complexity in Python, Java, JavaScript, and C#. A qualitative comparative methodology was adopted: a literature review was conducted in Scopus, IEEE Xplore, and Google Scholar (2018–2024); five code snippets were selected from public GitHub repositories; and a standardized prompt was applied to each AI. Experienced developers with over three years of experience then assessed the refactored outputs using a 1–5 Likert scale. The results demonstrated that each AI's performance in a given language is directly linked to the data corpus used during its training. It was concluded that the ideal choice of AI depends on both the target programming language and the specific development context.

Keywords: Refactoring;Generative Artificial Intelligence; Comparative Evaluation; ChatGPT; Perplexity; Gemini; GitHub Copilot; Bing.

1 INTRODUÇÃO

O avanço das inteligências artificiais generativas tem transformado significativamente o desenvolvimento de software, especialmente no contexto da refatoração de códigos (AlOmar et al., 2024a). A refatoração é uma prática essencial para manter a qualidade e a sustentabilidade de um projeto ao longo do tempo, pois melhora a estrutura e a legibilidade do código sem alterar sua funcionalidade(Fowler, 2018), ao eliminar redundâncias e duplicações, diminui as chances de inconsistências e facilita a manutenção.

Com menos código repetido, qualquer alteração necessária pode ser feita em um único ponto, otimizando o esforço dos desenvolvedores (White et al., 2023; Oliveira; Keuning; Jeuring, 2023). Porém, esse processo pode ser demorado e exigir um esforço significativo dos desenvolvedores; Estudos indicam que entre 20% e 30% do tempo de desenvolvimento pode ser gasto em refatoração manual (Almogahed et al., 2023). Ferramentas como ChatGPT, GitHub Copilot, Bing, Gemini e Perplexity surgiram como soluções potenciais para automatizar esse processo, reduzindo o tempo e o esforço dos desenvolvedores.

No entanto, ainda existem dúvidas sobre a eficácia dessas ferramentas na refatoração de códigos. O problema central desta pesquisa

é avaliar qual delas apresenta melhor desempenho em termos de qualidade do código gerado e conformidade com boas práticas de engenharia de software. A automatização da refatoração, pode proporcionar ganhos substanciais de eficiência, tornando o tema relevante tanto para a ciência da computação quanto para a indústria de software (Park et al., 2024; Purwoko et al., 2023).

Diante desse cenário, este estudo tem como objetivo geral comparar os resultados das refatorações de código realizadas por inteligências artificiais generativas, identificando qual ferramenta é mais eficiente para automatizar esse processo. Para alcançar esse objetivo, os objetivos específicos definidos foram: selecionar inteligências artificiais generativas de uso gratuito e popular; aplicar essas ferramentas à tarefa de refatoração de códigos em diferentes linguagens de programação; e comparar os resultados gerados com base em critérios avaliativos práticos, utilizando uma análise feita por profissionais com mais de 3 anos de experiencia em programação, por meio de uma escala Likert.

Para alcançar o propósito deste estudo, as ferramentas, Gemini, Bing, GitHub Copilot, ChatGPT e Perplexity, foram cuidadosamente examinada em detalhes, a revisão bibliográfica foi realizada em bases de dados como Scopus, IEEE e Google Scholar, abrangendo artigos publicados entre 2018 e 2023. O ChatGPT, mostra-se extremamente versátil graças ao seu treinamento em vastos volumes de texto, sendo capaz de responder dúvidas, redigir artigos, revisar trechos de código e até gerar composições artísticas, o que reflete seu impacto em áreas que vão do atendimento ao cliente à educação e à saúde (Purwoko et al., 2023; Guo, 2023; AlOmar et al., 2024a).

Inspirado por essa capacidade, o Bing Chat incorpora o GPT-4 diretamente ao navegador Microsoft Edge, funcionando como um copiloto que registra o histórico de conversas, fornece imagens geradas a partir de descrições textuais e até formula hipóteses embasadas em dados reais, exibindo sempre as fontes utilizadas para maior transparência (King, 2023; Al-Janabi; Alyasiri; Jebur, 2023). No mesmo ano de 2023, o Google lançou o Gemini para suceder o Bard e desafiar o GPT da OpenAI, adotando uma arquitetura multimodal que processa e gera texto, imagens, gráficos e diversos outros formatos, integrando-se ao buscador e a outras plataformas da empresa para aprimorar a relevância e a profundidade das respostas (Mcintosh et al., 2023; Raut; Katti, 2024; Islam; Ahmed, 2024).

Paralelamente, o Perplexity emergiu como alternativa competi-

tiva ao oferecer não apenas integração de imagens e vídeos, mas também um mecanismo de busca refinado que coleta dados em tempo real, recomenta aprimoramentos de consultas via recurso “Copilot” e mantém a privacidade do usuário ao dispensar login e coleta de dados pessoais (Perplexity AI, 2024; Perplexity Team, 2024). Por fim, o GitHub Copilot, baseado no modelo Codex da OpenAI, atua como um “par-programador” dentro do Visual Studio Code, gerando sugestões de código, correções de bugs e testes automáticos em tempo real, acelerando tarefas repetitivas e servindo como importante instrumento de aprendizado para desenvolvedores de diferentes níveis (Wermelinger, 2023; Puryear; Sprint, 2022).

Com este estudo, busca-se contribuir para a área de computação ao demonstrar a eficácia das IAs na refatoração de códigos, auxiliando desenvolvedores e empresas na escolha da ferramenta mais adequada para otimizar seus processos.

O trabalho está estruturado da seguinte forma: A primeira seção apresenta o problema de pesquisa, justifica a escolha do tema, expõe os objetivos geral e específicos e apresenta a fundamentação teórica; a segunda seção apresenta os artigos utilizados como base para a pesquisa, destacando suas limitações em relação ao tema abordado; A terceira seção descreve detalhadamente todo o processo de desenvolvimento da pesquisa, desde a escolha do tema até a análise dos resultados. Já quarta seção, são discutidos os resultados obtidos, analisando sua relevância em comparação com trabalhos relacionados e apresentando as limitações identificadas; Por fim, a conclusão expõe os principais achados do estudo em relação aos objetivos propostos, além de indicar as contribuições e sugestões para pesquisas futuras.

2 TRABALHOS CORRELATOS

Na área da computação, observam-se diversos estudos que avaliam isoladamente o desempenho de IAs na refatoração de códigos, por exemplo, o *Estudo de Caso: ChatGPT-3.5 em LeetCode e BeeCrowd* apresentou o desempenho do ChatGPT-3.5 em cem desafios das plataformas LeetCode e BeeCrowd, medindo apenas a taxa de acertos sem avaliar a qualidade do código ou comparar outras IAs (Souza, 2023). Em *How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations*, foram analisadas mais de 17 000 interações entre desenvolvedores e ChatGPT para identificar padrões de solicitação em refatoração, mas sem mensurar objetivamente a qualidade do resultado nem

incluir múltiplas IAs na comparação (AlOmar et al., 2024b).

O estudo *AI-Assisted Coding: Experiments with GPT-4*, da Universidade de Stanford, investigou o GPT-4 em codificação interativa, refatoração e geração de testes, evidenciando a necessidade de ajustes humanos, porém restringindo-se a esse único modelo (Beguš; Lu; Poldrack, 2023). Por fim, *Code Correctness and Quality in the Era of AI Code Generation – Examining ChatGPT and GitHub Copilot* comparou ChatGPT e Copilot em Java, usando métricas estáticas de qualidade, mas avaliou apenas duas IAs e uma única linguagem (Hansson; Ellréus, 2023).

Em contraste, o presente estudo amplia o escopo ao comparar cinco IAs gratuitas, ChatGPT, GitHub Copilot, Bing, Gemini e Perplexity, em quatro linguagens (Python, Java, JavaScript e C#), aplicando o mesmo prompt a todas e coletando avaliações anônimas de profissionais experientes por meio de escala Likert. Além de apresentar médias de desempenho, investigamos como a base de dados de treinamento influencia cada modelo, explicando por que certas ferramentas se destacam em linguagens específicas. Dessa forma, essa pesquisa oferece um panorama mais completo e fundamentado, fornecendo subsídios práticos para a escolha da IA mais adequada a cada contexto de desenvolvimento.

3 MATERIAIS E MÉTODOS

A escolha do tema deste estudo surgiu a partir de uma demanda recorrente na área da computação: a necessidade de entender qual ferramenta de inteligência artificial entrega melhores resultados na refatoração de códigos. Com o crescimento do uso dessas tecnologias entre estudantes e profissionais da área, tornou-se relevante investigar quais IAs realmente se destacam nessa tarefa. Para isso, foi adotada uma abordagem metodológica qualitativa.

As ferramentas selecionadas, GitHub Copilot, Gemini 2.5 Pro, Bing, Perplexity e ChatGPT GPT-4o, foram escolhidas com base em uma pesquisa prévia que considerou apenas aquelas disponíveis em versões gratuitas, populares e que oferecessem recursos voltados ao apoio em tarefas de programação.

Inicialmente, procedeu-se uma revisão bibliográfica aprofundada sobre cada uma das ferramentas escolhidas. Em seguida, deram-se início aos testes práticos por meio de experimentos independentes, incluindo geração e revisão de trechos de código, para verificar o desempenho de cada IA. Posteriormente, foram selecionados códigos em quatro das linguagens

mais utilizadas no mercado, Python, Java, JavaScript e C#, conforme o Índice TIOBE (junho de 2025), extraídos de repositórios públicos no GitHub, plataforma de acesso livre e sem burocracia. A seleção seguiu critérios de nível de dificuldade (fácil e médio) e extensão reduzida, de modo a garantir que as IAs tivessem acesso a trechos de código completos, condição essencial para maximizar a qualidade das sugestões de refatoração.

A refatoração dos códigos ocorreu em todas as ferramentas na versão gratuita, para verificar o desempenho acessível a qualquer usuário. Adotou-se como padrão de solicitação às IAs o seguinte comando no prompt: “Realize a refatoração do código a seguir, visando otimizar sua performance."

Com as versões refatoradas em mãos, elaborou-se um questionário no Google Forms, no qual cada trecho original aparecia acompanhado de suas cinco versões revisadas. Para assegurar a imparcialidade, as ferramentas foram identificadas apenas por letras, Ferramenta A, B, C, D e E, preservando-se o sigilo de sua identidade real.

Antes de iniciar a avaliação, os participantes informaram seu tempo de experiência profissional e as linguagens de programação com as quais tinham maior familiaridade, de modo que cada avaliador analisou apenas códigos na sua própria stack de domínio. O formulário foi enviado a professores e estudantes do Curso de Ciência da Computação da Unesc, obtendo-se 22 respostas válidas. Dessas, quatro respostas foram desconsideradas, pois foram fornecidas por desenvolvedores com menos de três anos de experiência, critério adotado para assegurar uma análise mais criteriosa.

Na segunda parte do questionário, os avaliadores atribuíram notas à qualidade de cada refatoração por meio de uma escala Likert que varia de 1 (não atende aos critérios básicos) a 5 (atende plenamente aos critérios). O formulário também incluía uma questão para avaliar o nível de dificuldade dos trechos de código antes da refatoração; entretanto, verificou-se que essa variável não exerceu influência significativa nos resultados das IAs, dado que todos os códigos se situavam nos níveis fácil e médio. Por essa razão, tais dados foram descartados, a fim de não desviar o foco da comparação entre as versões refatoradas.

Em seguida, todas as respostas foram exportadas para uma planilha do Excel, onde foi organizado os dados e calculados as médias de cada ferramenta em cada linguagem, procedimento que permitiu comparar de forma clara o desempenho comparativo das IAs.

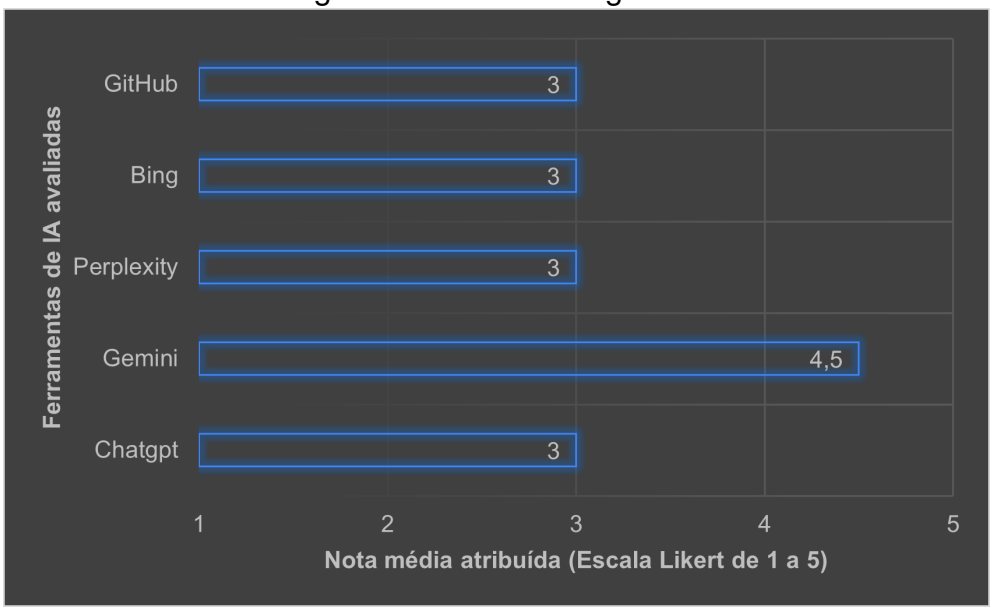
Finalmente, esses valores médios foram representados graficamente, tornando os resultados mais claros e fáceis de entender. Os gráficos foram organizados por linguagem de programação, possibilitando a avaliação do desempenho de cada ferramenta em cada uma das linguagens.

4 DISCUSSÃO E RESULTADOS

No caso do código Java (Figura 1), o Gemini se destacou com média de 4,5, mostrando que entende de forma consistente os padrões e boas práticas da linguagem; É importante mencionar que essa média foi feita a partir de 2 respostas válidas. Já o GitHub Copilot, o Bing, o Perplexity e o ChatGPT ficaram empatados com nota 3,0, embora tenham apresentado resultados satisfatórios, seu desempenho foi bem parecidos entre si.

Esse cenário sugere que o Gemini tem uma afinidade maior com a estrutura e a semântica do Java, enquanto as demais IAs entregam níveis semelhantes de refatoração, sem, contudo, alcançar o nível avançado do Gemini.

Figura 1 - Gráfico código Java



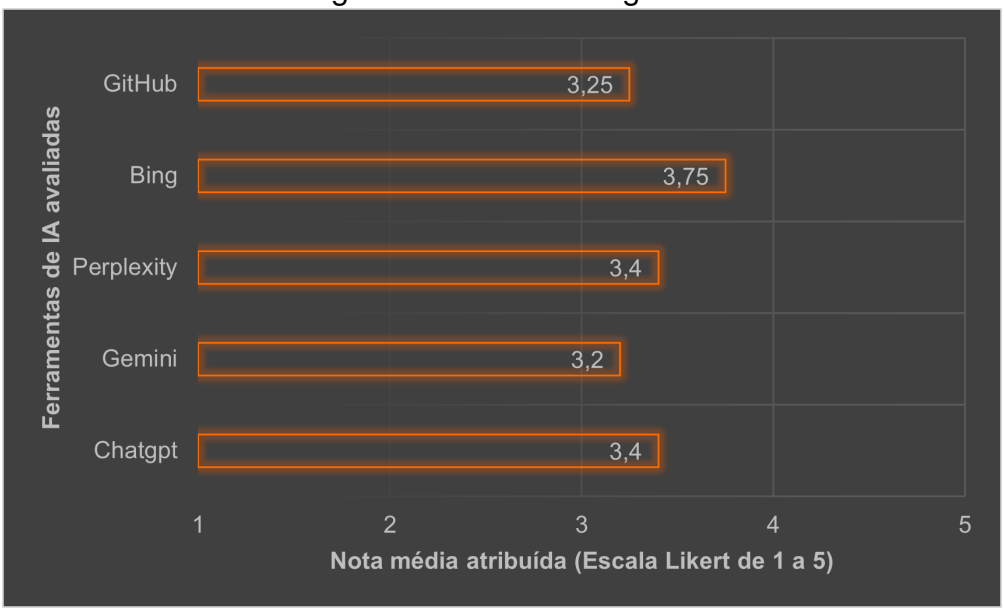
Fonte: Elaborado pelo autor.

Na linguagem C# (Figura 2), a situação muda um pouco, o Bing saiu na frente, com média de 3,75 , mostrando uma performance um pouquinho melhor. Logo atrás, empatados na segunda colocação, vêm o ChatGPT e o Perplexity, ambos com média de 3,4. O GitHub Copilot ficou com 3,25,

enquanto o Gemini fechou a lista com 3,2. Vale dizer que todas essas médias vieram de 6 respostas válidas.

Esses resultados revelam que o Bing, mesmo em sua versão gratuita, oferece sugestões de refatoração mais alinhadas às expectativas dos avaliadores em C#, ao passo que ChatGPT e Perplexity sustentam desempenho sólido e equilibrado. Copilot e Gemini, por sua vez, ainda que apresentem médias superiores a 3,0, demonstram maior espaço para aprimoramento nesse contexto.

Figura 2 - Gráfico código C#

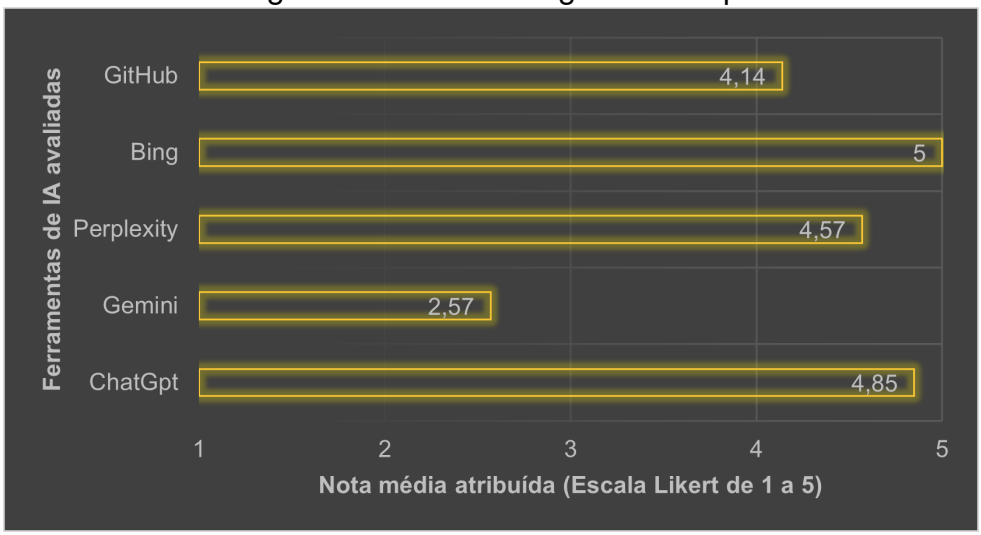


Fonte: Elaborado pelo autor.

Na refatoração de código JavaScript (Figura 3), o Bing se destacou por atingir a nota máxima 5,0, assumindo o primeiro lugar. Em seguida, o ChatGPT obteve média de 4,85, ficando muito próximo do desempenho do Bing. O Perplexity alcançou 4,57, e o GitHub Copilot, 4,14, ambos demonstrando boa capacidade de gerar código otimizado. Por fim, o Gemini apresentou a média mais baixa 2,57, indicando maior dificuldade em lidar com a estrutura do JavaScript. Os resultados apresentados foram calculados com base em 7 respostas válidas.

Os resultados demonstra que Bing e ChatGPT se destacam na habilidade de refatorar código de forma eficiente em JavaScript. Perplexity e Copilot também mostram competência satisfatória nessa tarefa, enquanto Gemini ainda necessita de melhorias significativas para alcançar um desempenho comparável.

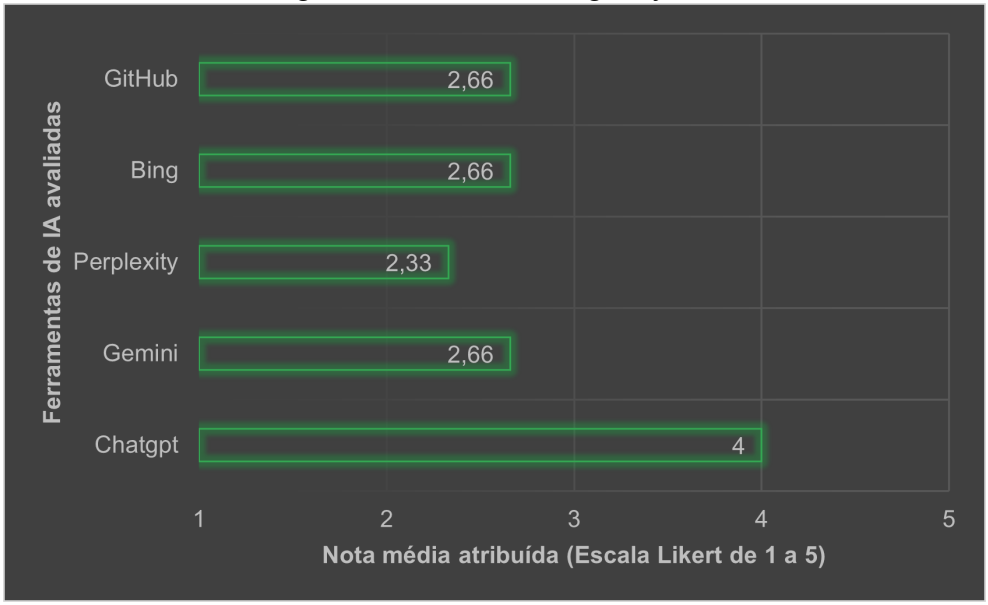
Figura 3 - Gráfico código JavaScript



Fonte: Elaborado pelo autor.

Na refatoração em Python (Figura 4), o ChatGPT se destacou com uma média exata de 4,0, demonstrando um desempenho sólido. Já o GitHub Copilot, Bing e Gemini apresentaram resultados semelhantes, empatando com média de 2,66. O Perplexity, por sua vez, obteve 2,33, ficando um pouco abaixo dos demais. Essas médias foram calculadas com base em 3 respostas válidas.

Figura 4 - Gráfico código Python

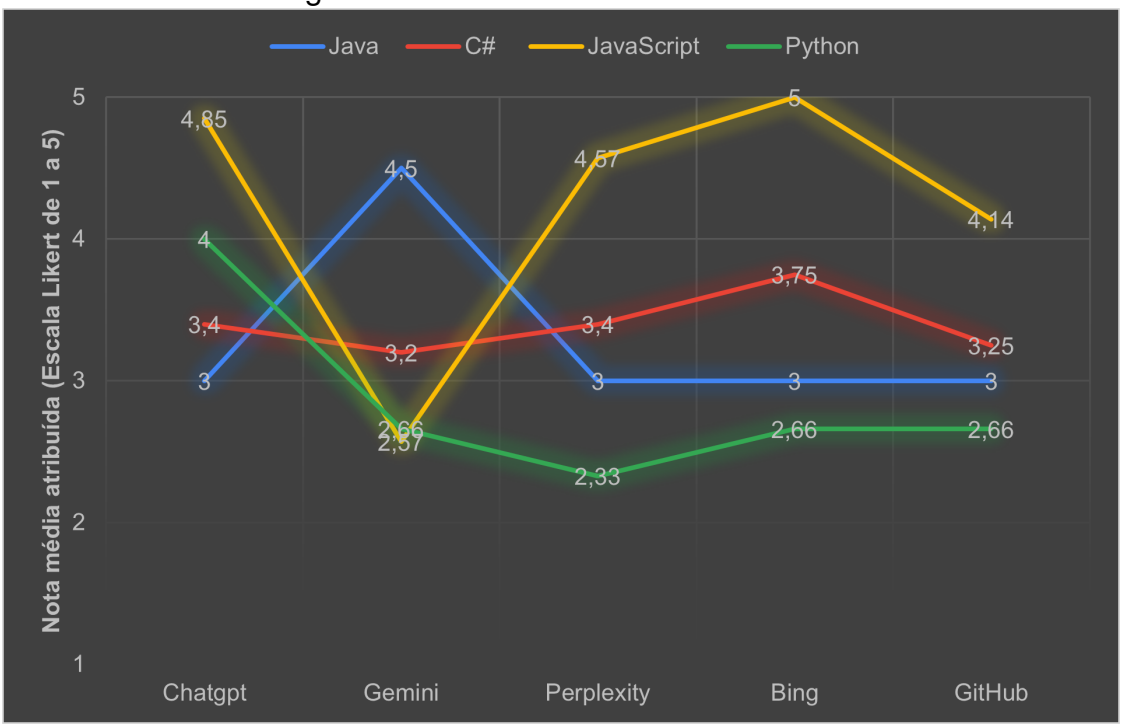


Fonte: Elaborado pelo autor.

Na média geral das avaliações (Figura 5), o ChatGPT sobressaiu com nota de 3,81, evidenciando sua capacidade de adaptação a dife-

rentes linguagens e consistência na geração de refatorações de qualidade. Em seguida, o Bing alcançou 3,60, demonstrando bom desempenho, ainda que tenha apresentado desafios em padrões de código mais complexos. O Perplexity (3,32) e o GitHub Copilot (3,26) ocuparam posições intermediárias, com resultados muito próximos, o que sugere que ambas oferecem refatorações eficazes, mas que ainda podem ser aprimoradas para rivalizar com as líderes. O Gemini, por fim, obteve a menor média (3,23), indicando menor aderência às boas práticas e menor receptividade por parte dos avaliadores. Esses achados destacam as forças e fraquezas relativas de cada ferramenta e ressaltam a importância de selecionar a IA de refatoração de acordo com as necessidades específicas do projeto e a complexidade do código-fonte.

Figura 5 - Média das Notas das IA's



Fonte: Elaborado pelo autor.

Embora pesquisas anteriores tenham se concentrado em um ou dois modelos de IA em contextos restritos, este estudo diferencia-se ao adotar uma abordagem comparativa envolvendo cinco ferramentas aplicadas a quatro linguagens de programação. Esse escopo ampliado oferece uma visão mais abrangente do desempenho de cada solução em cenários diversos, superando as limitações de alcance dos trabalhos prévios e fornecendo subsídios concretos para que desenvolvedores escolham a ferra-

menta de refatoração mais adequada às necessidades específicas de seus projetos.

5 CONCLUSÃO

Com base nos resultados obtidos, foi possível observar que cada ferramenta de inteligência artificial apresentou pontos fortes específicos em determinadas linguagens de programação. O Gemini, por exemplo, demonstrou uma afinidade maior com Java, atingindo nota 4,5 o que pode torná-lo uma boa opção, para desenvolvedores que trabalham principalmente com essa linguagem. No caso do C#, o Bing teve um excelente desempenho, com nota 3,75, possivelmente favorecido pelo fato de tanto a linguagem quanto a IA serem produtos da Microsoft. O Bing também se destacou no JavaScript, liderando as avaliações com nota 5,0, o que reforça seu potencial para desenvolvedores que utilizam essas duas tecnologias.

Observou-se que a performance das IAs está diretamente ligada às bases de dados utilizadas no treinamento de cada modelo. Por exemplo, a familiaridade da Google com Java e o forte vínculo da Microsoft com C# e JavaScript influenciaram os resultados obtidos pelo Bing, já que as IAs tendem a ter melhor desempenho nas linguagens para as quais foram mais expostas a exemplos de qualidade durante o pré-treinamento.

Já em Python, o ChatGPT apresentou o melhor desempenho entre todas as ferramentas, com nota 5,0, entregando refatorações bem estruturadas e claras. Além disso, o ChatGPT mostrou-se a ferramenta mais versátil, mantendo boas avaliações em todas as linguagens testadas. Essa consistência faz com que seja uma escolha sólida para desenvolvedores que atuam em projetos diversos ou que buscam uma solução equilibrada para múltiplos contextos de desenvolvimento.

Esses resultados reforçam a importância de considerar o tipo de linguagem antes de escolher uma IA para auxiliar na refatoração de código. A pesquisa, portanto, oferece dados práticos que ajudam na tomada de decisão e pode servir como base para estudos futuros.

Este estudo supriu a lacuna existente na literatura ao realizar uma avaliação comparativa e sistemática de cinco inteligências artificiais em quatro linguagens de programação, identificando quais ferramentas oferecem as refatorações de código mais eficientes e alinhadas às boas práticas de engenharia de software, além de fornecer subsídios práticos para a escolha da IA mais adequada a cada contexto de desenvolvimento.

Para trabalhos futuros, as possibilidades são diversas. Além de

ampliar a amostra de avaliadores, seria interessante explorar as ferramentas avançadas já disponíveis em cada IA, por exemplo, recursos que permitem “pensar” por mais tempo antes de retornar uma resposta ou configurações de ajuste fino voltadas a refatoração de códigos. Incluir a medição do tempo de processamento de cada sistema também pode oferecer percepções interessantes.

Outra alternativa interessante é comparar em detalhe as limitações que cada ferramenta apresenta durante a refatoração, identificando pontos em que uma falha ou demora mais que outra. Ademais, testar linguagens adicionais (como Go, Rust, Kotlin, TypeScript...) ajudaria a verificar se os padrões de desempenho observados em Java, C#, Python e JavaScript se repetem em outros contextos de programação. Por fim, recomenda-se incluir, além das versões gratuitas, opções pagas, e as novas soluções que venham a surgir no mercado.Essas abordagens podem proporcionar uma nova perspectiva sobre o uso das IAs generativas na refatoração de códigos.

REFERÊNCIAS

AL-JANABI, O. M.; ALYASIRI, O. M.; JEBUR, E. A. **GPT-4 versus Bard and Bing: LLMs for Fake Image Detection**. 2023. 249-254 p.

ALMOGAHED, A. et al. **A Refactoring Classification Framework for Efficient Software Maintenance**. 2023. 78904-78917 p.

ALOMAR, E. A. et al. **How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations**. 2024. 202-206 p.

ALOMAR, E. A. et al. **How to Refactor this Code? An Exploratory Study on Developer-ChatGPT Refactoring Conversations**. 2024. arXiv–2402 p.

BEGUš, G.; LU, T.; POLDRACK, R. A. **Codificação assistida por IA: Experimentos com GPT-4**. 2023. Acesso em: 28 de outubro de 2024. Disponível em: <https://arxiv.org/pdf/2304.13187>.

FOWLER, M. **Refactoring: improving the design of existing code**. [S.l.]: Addison-Wesley Professional, 2018.

GUO, P. J. **Six Opportunities for Scientists and Engineers to Learn Programming Using AI Tools Such as ChatGPT**. 2023. 73-78 p.

HANSSON, E.; ELLRéUS, O. Bachelor’s thesis, **Code Correctness and Quality in AI-Assisted Coding: Experiments with ChatGPT and GitHub Copilot**. 2023. Supervisor: Welf Löwe.

ISLAM, R.; AHMED, I. **Gemini-the most powerful LLM: Myth or Truth**. 2024. 303-308 p.

KING, M. **Bing chatbot formulating and testing novel hypotheses in real-time: How slime, chocolate, and Nobel prizes reveal the power and limits of artificial intelligence**. [S.l.]: Engineering Archive, 2023.

MCINTOSH, T. R. et al. **From google gemini to openai q*(q-star): A survey of reshaping the generative artificial intelligence (ai) research landscape**. 2023.

OLIVEIRA, E.; KEUNING, H.; JEURING, J. **Student code refactoring misconceptions**. 2023. 19–25 p.

PARK, D. et al. **A Study on Performance Improvement of Prompt Engineering for Generative AI with a Large Language Model**. 2024. 1187–1206 p. Disponível em: <https://journals.riverpublishers.com/index.php/JWE/article/view/24479>.

Perplexity AI. **Getting Started with Perplexity**. 2024. Disponível em: <https://www.perplexity.ai/hub/getting-started>.

Perplexity Team. **Introducing Perplexity Pages**. 2024. Disponível em: <https://www.perplexity.ai/hub/blog/perplexity-pages>.

PURWOKO, J. W. et al. **Analysis ChatGPT Potential: Transforming Software Development with AI Chat Bots**. 2023. 36-41 p.

PURYEAR, B.; SPRINT, G. **Github copilot in the classroom: learning to code with AI assistance**. [S.l.]: Consortium for Computing Sciences in Colleges, 2022. 37–47 p.

RAUT, A.; KATTI, J. **The Future of Chatbots: Survey on Generative Pretrained Models, Ernie, and Gemini**. 2024. 1-5 p.

SOUZA, D. L. de L. **TCC Artigo Ciência da Computação**. 2023. Acesso em: 5 nov. 2024. Disponível em: <http://dspace.sti.ufcg.edu.br:8080/jspui/bitstream/riufcg/30461/1/D%C3%89BORA%20L%C3%8aDA%20DE%20LUCENA%20SOUZA%20-%20TCC%20ARTIGO%20CI%C3%8aNCIA%20DA%20COMPUTA%C3%87%C3%83O%20CEEI%202023.pdf>.

WERMELINGER, M. **Using github copilot to solve simple programming problems**. 2023. 172–178 p.

WHITE, J. et al. **ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design**. 2023. Disponível em: <https://arxiv.org/abs/2303.07839>.